จเร เลิศสุดวิชัย

เทคนิคการ เพิ่มความเร็ว ให้คลิปเปอร์

รวบรวมเทคนิควิธีการเขียนโปรแกรมคลิปเปอร์เพื่อให้ได้โปรแกรมที่ทำงานได้เร็วขึ้น

คลิปเปอร์ 5.2 ได้ปรับปรุงภาษาของตัวเองให้มีรูปแบบและการทำงานที่ใกล้เคียงภาษาระดับทั่วไปมากขึ้น โดยเฉพาะหากเปรียบเทียบกับภาษาซีจะพบว่ามีหลายอย่างที่คลิปเปอร์ทำตามแบบภาษาซี ดังนั้นหากผู้ที่เคยเขียนโปรแกรมภาษาซีมาก่อนแล้วมาเขียนคลิปเปอร์จะทำให้สามารถเข้าใจการทำงานของคลิปเปอร์ได้เร็ว แต่หากเป็นผู้ที่เคยเขียนโปรแกรมคลิปเปอร์ในรุ่นเก่าคือ Summer'87 มาก่อนแล้วหันมาใช้คลิปเปอร์ 5.2 แทน ซึ่งท่านอาจจะใช้รูปแบบภาษาแบบเดิมใน Summer'87 ก็ได้ เพราะคลิปเปอร์ 5.2 ยังคงสนับสนุนอยู่ แต่หากท่านจะใช้รูปแบบโครงสร้างภาษาใหม่ในรุ่น 5.2 จะต้องศึกษาและทำความเข้าใจภาษาใหม่เกือบทั้งหมด แต่ในการเขียนโปรแกรมบางครั้งหากติดใจในรูปแบบเดิมแล้วนำมาใช้อีกซึ่งรูปแบบภาษาเดิมจะเป็นสิ่งที่ทำให้โปรแกรมที่ได้มีการทำงานช้า ดังนั้นในบทความนี้ผู้เขียนได้รวบรวมเทคนิควิธีการเขียนโปรแกรมคลิปเปอร์เพื่อให้ได้โปรแกรมที่ทำงานได้เร็วขึ้น

เทคนิคที่ 1

ให้พยายามใช้ตัวแปรแบบโลคัล (Local) และสแตติก (Static) แทนตัวแปรแบบพายเวต (Private) และพับบลิค (PUBLIC)

ตัวแปรแบบพายเวตและพับบลิคเป็นตัวแปรที่มีมาแต่เดิมคือเริ่มมาตั้งแต่ในภาษา XBASE ของโปรแกรมดีเบส ซึ่งทำให้นักเขียนโปรแกรมหลายคนมักจะใช้ตัวแปรทั้งสองเสมอ แต่จากนี้ต่อไปหากท่านใดที่ใช้งานตัวแปรทั้งสอง

ชนิดอยู่ ขอให้ระมัดระวังการใช้งาน โดยพยายามใช้งานให้น้อยลง หรือใช้เท่าที่จำเป็นเท่านั้น (เพราะในบางครั้งก็ต้องใช้ตัวแปรพายเวดและพับบลิค เนื่องจากการทำงานบางอย่างไม่สามารถกระทำกับตัวแปรโลคัลและสแตติกได้ โดยเฉพาะในเรื่องของมาโคร)

ตัวแปรพายเวดและพับบลิคจะเกิดขึ้นเมื่อโปรแกรมทำงาน โดยคลิปเปอร์จะสร้างตารางสำหรับเก็บรายชื่อ (Symbol tables) ของตัวแปรทั้งสองชนิด ซึ่งหากในโปรแกรมมีการใช้งานหรืออ้างถึงชื่อตัวแปรทั้งสองระหว่างการทำงาน คลิปเปอร์จะทำการตรวจสอบชื่อตัวแปรจากตารางเก็บรายชื่อ เมื่อพบชื่อตัวแปรในตารางสัญลักษณ์ ก็จะทราบที่เก็บข้อมูล (Address) ของตัวแปร ดังนั้นจะเห็นว่าหากในโปรแกรมมีการใช้งานตัวแปรชนิดพายเวดและพับบลิค โปรแกรมดังกล่าวจะมีการทำงานที่ช้า เพราะเนื่องจากขั้นตอนของการค้นชื่อในตารางสัญลักษณ์นั่นเอง

หากสรุปตรงนี้จะเห็นว่าตัวแปรพายเวดและตัวแปรพับบลิคมีข้อเสียอย่างเห็นชัดเจนคือ ทำให้โปรแกรมช้าและยังสิ้นเปลืองหน่วยความจำในขณะที่โปรแกรมกำลังทำงานด้วย เพราะต้องใช้พื้นที่หน่วยความจำบางส่วนไปในการสร้างตารางสัญลักษณ์เก็บชื่อตัวแปร ข้อเสียอีกจุดหนึ่งซึ่งผู้ที่เขียนโปรแกรมคลิปเปอร์ในรุ่น Summer'87 จะทราบดีคือเรื่องของขอบเขตหรือการมองเห็น (scope) ของตัวแปรทั้งสอง โดยตัวแปรพับบลิคสามารถมองเห็นหรือถูกอ้างใช้งานได้ตลอดโปรแกรม ส่วนตัวแปรพายเวดซึ่งถึงแม้ชื่อชนิดตัวแปรจะมีความหมายว่าส่วนตัว แต่ความจริงแล้วขอบเขตหรือการถูกมองเห็นของมันกลับไม่เป็นเช่นชื่อ คือสามารถมองเห็นและอ้างใช้งานได้ในโปรแกรมย่อย (ฟังก์ชันหรือโพรซีเดอร์) ที่ถูกเรียกจากโปรแกรมย่อยที่ประกาศตัวแปรพายเวดเอาไว้ ซึ่งขอบเขตตัวแปรนี้เป็นเรื่องใหญ่สำหรับนักเขียนโปรแกรม เพราะจะทำให้โปรแกรมย่อยหรือโมดูลโปรแกรมแต่ละส่วนไม่สามารถแยกกันอย่างอิสระได้ หรือจะเรียกว่าขาดความเป็นโมดูล่าร์ (Modular) ไปนั่นเอง ซึ่งจะทำให้โปรแกรมเกิดบัก (Bug) ได้ง่าย

จากข้อเสียทั้งหมดนี้คลิปเปอร์ 5.2 จึงได้เสนอชนิดของตัวแปรเพิ่มขึ้นมาคือ โลคัล (Local) และสแตติก (Static) เพื่อที่จะให้ผู้เขียนโปรแกรมได้ใช้แทนตัวแปรพายเวดและพับบลิค โดยตัวแปรใหม่ทั้งสองมีคุณสมบัติที่สามารถแก้ไขข้อเสียของตัวแปรพายเวดและพับบลิคได้ เช่นในเรื่องของตำแหน่งที่อยู่ ในตัวแปรโลคัลและสแตติกนั้นจะมีที่อยู่ทันทีเมื่อโปรแกรมถูกแปลหรือคอมพายล์ (ซึ่งโดยทั่วไปจะเรียกช่วงเวลาที่โปรแกรมถูกแปลว่า Compile Time ส่วนช่วงเวลาขณะโปรแกรมทำงานจะเรียกว่า Run Time) ดังนั้นเมื่อโปรแกรมทำงานในคำสั่งที่มีการอ้างถึงตัวแปรทั้งสอง โปรแกรมจะสามารถรู้ถึงที่อยู่ของข้อมูลในตัวแปรทันที โดยไม่ต้องเสียเวลาในการค้นหาจากตารางสัญลักษณ์อีกเหมือนตัวแปรพายเวดและพับบลิค ซึ่งจะทำให้โปรแกรมมีการทำงานเร็วขึ้น โดยเร็วขึ้นถึง 2 เท่าของโปรแกรมที่ใช้ตัวแปรพายเวดและพับบลิค

มิใช่เพียงแต่ในเรื่องของความเร็วของโปรแกรมที่เพิ่มขึ้นเท่านั้น การใช้งานตัวแปรโลคัลและสแตติกยังช่วยให้โปรแกรมแต่ละส่วนมีการแยกกันอย่างเด่นชัดขึ้น หรือมีความเป็นโมดูล่าร์นั่นเอง ซึ่งเป็นเรื่องสำคัญและเป็นความต้องการของนักเขียนโปรแกรมทุกคน

ดังนั้นหากท่านผู้อ่านกำลังเขียนโปรแกรมด้วยคลิปเปอร์อยู่ละก็ ท่านควรหลีกเลี่ยงการใช้ตัวแปรชนิดพายเวดและพับบลิคเสีย โดยเฉพาะกับโปรแกรมของท่านที่คิดว่าจะสร้างเก็บไว้เป็นไลบรารีเพื่อนำมาใช้ภายหลัง ยิ่งสมควรพิจารณาเป็นอย่างยิ่ง เพราะหากท่านหลีกเลี่ยงแล้วหันมาใช้ตัวแปรโลคัลและสแตติก จะทำให้โปรแกรมของท่านมีความถูกต้องในการทำงาน มีความน่าเชื่อถือมากขึ้น นอกจากนั้นยังทำให้ท่านสามารถทดสอบโปรแกรมได้ง่ายขึ้น โดยไม่ต้องกังวลใจว่าตัวแปรที่จะสร้างขึ้นมาใหม่จะไปซ้ำซ้อนกับตัวแปรในโมดูลอื่น ๆ และการนำเอาโปรแกรมหรือไลบรารีมาใช้งานก็จะสะดวกขึ้นด้วย

การเปลี่ยนตัวแปรพายเวดเป็นตัวแปรโลคัล

มีรูปแบบของตัวแปรพายเวดอยู่ 4 วิธี ดังเช่นฟังก์ชันต่อไปนี้

```
Function Example
        Parameter a, b, c
        Private x, y, z
        Declare aList [100]

        q : = 123
Return Nil
```

รูปแบบการประกาศตัวแปรพายเวดทั้ง 4 วิธีนี้ ขออธิบายแยกเป็นกรณี ๆ ไปว่าสามารถหลบเลี่ยงและสามารถใช้ตัวแปรโลคัลแทนได้อย่างไร

1. การประกาศพารามิเตอร์โดยใช้คำสั่ง PARAMETER ซึ่งพารามิเตอร์ที่เกิดขึ้นจากคำสั่งนี้จะมีคุณสมบัติแบบตัวแปรพายเวดทุกประการ ดังนั้นจึงควรเลี่ยงการประกาศพารามิเตอร์ด้วยวิธีนี้ โดยให้ใช้การประกาศพารามิเตอร์

ในรูปแบบใหม่ที่คลิปเปอร์ 5.2 เตรียมไว้คือ รูปแบบลิสต์ของพารามิเตอร์ (Parameter list) ซึ่งรูปแบบการประกาศเป็นดังนี้

```
Function Example (a, b, c)
```

พารามิเตอร์ที่เกิดขึ้นจากการประกาศนี้จะถูกมองเป็นตัวแปรแบบโลคัลโดยอัตโนมัติ

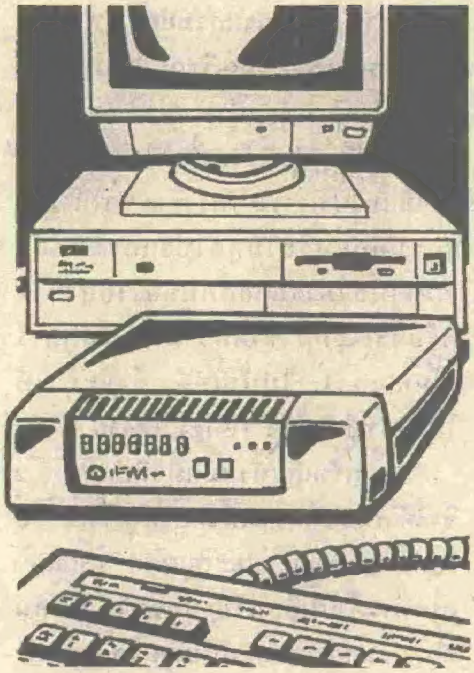
2. การประกาศตัวแปรด้วยคำสั่ง PRIVATE หากว่าไม่มีความจำเป็นจริง ๆ (บางครั้งจำเป็นต้องใช้ เช่น ในกรณีที่มีการใช้งานตัวแปรในมาโคร) ควรเปลี่ยนมาใช้การประกาศด้วยคำสั่ง LOCAL แทน เช่น

```
Local x, y, z
```

3. การประกาศตัวแปรด้วยคำสั่ง DECLARE ซึ่งความจริงแล้วก็คือการประกาศตัวแปรด้วยคำสั่ง PRIVATE นั่นเอง ดังนั้นจึงควรใช้คำสั่ง LOCAL แทน ดังเช่นการประกาศตัวแปรอะเรย์ข้างต้นก็สามารถประกาศได้ใหม่ดังนี้

```
Local aList [100]
```

4. ตัวแปรที่ไม่ได้ประกาศ แต่อยู่ ๆ ก็มีการใช้งานเลย ตัวแปรนี้จะถูกสร้างขึ้นเป็นตัวแปรชนิดพายเวตโดยอัตโนมัติ เช่น ในตัวอย่างข้างต้นตัวแปร q นี้จะเป็นตัวแปรชนิดพายเวต ลักษณะเช่นนี้สามารถแก้ไขได้ง่าย โดยการประกาศตัวแปรนั้นด้วยคำสั่ง LOCAL เสีย การค้นหาตัวแปรลักษณะในโปรแกรมที่เขียนโดยไม่มีการประกาศนั้นสามารถทำได้โดยขณะแปลโปรแกรมคลิปเปอร์หรือคอมพายล์ให้ระบุการคอมพายล์ด้วย /W เมื่อพบข้อความเตือนจากตัวแปรลักษณะดังกล่าวก็ให้แก้ไข อย่างไรก็ตาม สิ่งเหล่านี้จะไม่เกิดขึ้นหากต่อไปในการเขียนโปรแกรม



ให้พยายามประกาศตัวแปรก่อนที่จะใช้ทุกครั้งเสมอไป โดยพยายามตั้งให้เป็นกฎของการเขียนโปรแกรม

สิ่งที่กล่าวมาแล้วข้างต้นจะช่วยให้ท่านเขียนโปรแกรมคลิปเปอร์ได้ดีนอกเหนือไปจากความเร็วของการทำงานของโปรแกรมที่เพิ่มขึ้น อย่างไรก็ตาม เป็นเรื่องยากทีเดียวที่จะทำการเปลี่ยนตัวแปรพายเวตเป็นตัวแปรโลคัลในโปรแกรมเก่าที่เคยเขียนมาก่อน โดยเฉพาะคลิปเปอร์รุ่นซัมเมอร์ 87 ซึ่งผู้ที่เขียนโปรแกรมจะทราบดีว่าโปรแกรมที่เขียนด้วยคลิปเปอร์รุ่นนี้ไม่มีความเป็นโมดูล่าร์ในตัวโปรแกรม หรือไม่สามารถแยกส่วนของโปรแกรมได้อิสระ ดังนั้นหากมีการเปลี่ยนแปลงตัวแปรก็อาจจะทำให้โปรแกรมนั้นไม่สามารถทำงานได้ดังเดิม แต่หากเป็นโปรแกรมใหม่ที่ท่านเริ่มต้นเขียนใหม่ก็ขอให้ท่านพยายามใช้ตัวแปรโลคัลแทนตัวแปรพายเวต แล้วท่านจะเดินถูกทางในการเขียนโปรแกรมโดยเดินไปสู่เส้นทางของโปรแกรมแบบโมดูล่าร์

การเปลี่ยนตัวแปรพับบลิคเป็นตัวแปรสแตติก

การเปลี่ยนตัวแปรพับบลิคเป็นตัวแปรสแตติกนั้นไม่สามารถทำได้ทันทีทันใดเหมือนกรณีเปลี่ยนตัวแปรพายเวตเป็นโลคัล โดยต้องพิจารณาจากการใช้งานของตัวแปรพับบลิคจากโปรแกรมเสียก่อน ทั้งนี้เพราะว่าตัวแปรพับบลิคและสแตติกมีข้อแตกต่างกันในเรื่องของขอบเขตของตัวแปร โดยตัวแปรพับบลิคนั้นสามารถถูกอ้างใช้ได้ตลอดทั้งโปรแกรมแม้ว่าจะอยู่ในโปรแกรมต้นฉบับคนละไฟล์ก็ตาม แต่ขอบเขตของตัวแปรสแตติกจะมี 2 ลักษณะขึ้นอยู่กับการประกาศ หากประกาศอยู่ในฟังก์ชันหรือโพรซีเดอร์ขอบเขตก็จะมีอยู่เพียงในฟังก์ชันหรือโพรซีเดอร์เท่านั้น แต่หากประกาศไว้นอกฟังก์ชันโพรซีเดอร์ โดยประกาศไว้ในไฟล์ต้นฉบับใด ขอบเขตของตัวแปรนั้นก็จะอยู่ภายในไฟล์ต้นฉบับนั้น ซึ่งทุกฟังก์ชันโพรซีเดอร์ที่อยู่ในไฟล์ต้นฉบับดังกล่าวจะสามารถอ้างถึงและใช้งานตัวแปรได้ ดังนั้นการเปลี่ยนตัวแปรจากพับบลิคเป็นสแตติกจึงต้องพิจารณาจากวัตถุของการใช้งานตัวแปรพับบลิคเป็นหลัก โดยสามารถแบ่งการพิจารณาออกเป็น 3 ลักษณะคือ

1. การใช้งานตัวแปรพับบลิคเพื่อเก็บค่าข้อมูลของการทำงานครั้งก่อน โดยการใช้ตัวแปรนี้อยู่ในฟังก์ชันหรือโพรซีเดอร์เดียว หากผู้ที่เคยเขียนโปรแกรมด้วยคลิปเปอร์ซัมเมอร์ 87 จะทราบว่า ตัวแปรพับบลิคนั้นเมื่อประกาศขึ้นแล้วจะคงมีชีวิตหรือช่วงชีวิต (Life Time) ยาวนานจนกว่าโปรแกรมจะจบการทำงานลง หมายความว่าตัวแปรพับบลิคมีการใช้งานแล้ว ตัวแปรดังกล่าวจะยังคงเก็บค่าข้อมูลเดิมเอาไว้

และสามารถถูกอ้างใช้ได้อีกภายหลัง ซึ่งลักษณะช่วงชีวิตเหล่านี้ก็มีอยู่ในตัวแปรสแตติก ดังนั้นการเปลี่ยนตัวแปรพับบลิคในกรณีนี้สามารถทำได้ไม่ยากนัก โดยหากว่ามีเพียงฟังก์ชันหรือโพรซีเดอร์เดียวที่ใช้งานตัวแปรดังกล่าวก็ให้เปลี่ยนมาใช้ตัวแปรสแตติกในลักษณะภายใน (Internal Static) คือประกาศตัวแปรไว้ภายในฟังก์ชันโพรซีเดอร์นั่นเอง ดังตัวอย่างในรูปที่ 1 และรูปที่ 2

ในรูปที่ 1 แสดงฟังก์ชัน page__footer โดยเป็นกรณีของโปรแกรมที่เขียนในคลิปเปอร์ซัมเมอร์ 87 ส่วนในรูปที่ 2 เป็นโปรแกรมคลิปเปอร์ 5.2 ซึ่งจะมีการทำงานลักษณะเดียวกันสามารถแทนกันได้ โดยโปรแกรมในรูปที่ 2 นั้น ตัวแปร page จะไม่สามารถถูกอ้างใช้งานจากใครได้อีก

ลักษณะการทำงานของฟังก์ชันทั้งสองเหมือนกัน โดยฟังก์ชันนี้จะถูกเรียกใช้เมื่อมีการพิมพ์รายงานแต่ละครั้ง ซึ่งจะต้องพิมพ์หมายเลขหน้าของรายงานและหมายเลขจะต้องเพิ่มขึ้นเพื่อรอการพิมพ์ของหน้าถัดไป

2. กรณีที่ใช้ตัวแปรพับบลิคเพื่อให้ฟังก์ชันโพรซีเดอร์ภายในไฟล์โปรแกรมต้นฉบับเดียวกันเท่านั้นที่มองเห็นและเรียกใช้งานได้ ลักษณะนี้สามารถเปลี่ยนตัวแปรได้อย่างทันที โดยการประกาศตัวแปรแบบสแตติกแทนได้ที่ต้นไฟล์โปรแกรมต้นฉบับ (External Static) อย่างไรก็ตาม การประกาศตัวแปรสแตติกแบบนี้เมื่อคอมพายล์โปรแกรมจะต้องระบุข้อเลือกการคอมพายล์ด้วย /N ดังตัวอย่างรูปที่ 3 และรูปที่ 4

ในรูปที่ 3 ตัวแปร page นั้นถูกใช้ทั้งในฟังก์ชัน page__footer และ page__set ทั้งสองอยู่ในไฟล์โปรแกรมต้นฉบับเดียว ซึ่งโปรแกรมในรูปที่ 4 จะมีความเป็นอิสระจากไฟล์โปรแกรมอื่นมากกว่า ทั้งนี้เพราะตัวแปร page จะสามารถมองเห็นและถูกใช้งานได้เฉพาะภายในไฟล์ต้นฉบับในรูปที่ 4 เท่านั้น

3. กรณีที่ใช้ตัวแปรพับบลิคเพื่อให้ฟังก์ชันโพรซีเดอร์อื่น ๆ ที่กระจายอยู่ในไฟล์โปรแกรมต้นฉบับต่าง ๆ สามารถใช้งานตัวแปรได้ ลักษณะนี้ไม่สามารถจะเปลี่ยนเป็นตัวแปรสแตติกได้ทันที แต่จะต้องมีการเพิ่มเติมส่วนอื่นเข้าไปเสริมให้มีการทำงานเช่นเดิมได้ด้วย ซึ่งหากเปลี่ยนตัวแปรพับบลิคเป็นตัวแปรสแตติกแล้วจะต้องเพิ่มฟังก์ชันที่ทำฟังก์ชันโพรซีเดอร์ที่เก็บอยู่คนละไฟล์อื่นให้สามารถอ่านค่าและเปลี่ยนค่าได้ โดยจะต้องเข้าไปเปลี่ยนในฟังก์ชันหรือโพรซีเดอร์อื่นซึ่งก่อนหน้านี้เคยอ้างใช้ชื่อตัวแปรในการใช้ค่าและเปลี่ยนค่า ให้เป็นการเรียกผ่านฟังก์ชันที่เตรียมไว้ให้ ลักษณะตัวอย่างที่เห็นชัดเจนได้แก่ ฟังก์ชัน SETCOLOR( ) ซึ่งเป็นฟังก์ชันของคลิปเปอร์เอง ฟังก์ชันนี้สามารถอ่านเอาค่าของสีของตัวอักษรออกมาได้โดยการเรียกชื่อฟังก์ชันเฉย ๆ แต่หากต้องการเปลี่ยนค่าสีก็ให้ผ่านสีเข้าไปเป็นพารามิเตอร์ของฟังก์ชัน SETCOLOR( ) เช่น SETCOLOR ("W/B") เป็นต้น ซึ่งจากการทำงานดังกล่าว อาจเขียนเป็นโปรแกรมได้ดังรูปที่ 5

```
Function page_footer
   If Type ("page") = "U"
      Public page
      page = 0
   Else
      page = page + 1
   Endif
   ? footer_text + "Page " + Str (page,3)
   Eject
Return
*
```

รูปที่ 1 แสดงฟังก์ชัน page-footer กรณีที่เขียนในคลิปเปอร์ซัมเมอร์ 87

```
Function page_footer
   Static page := 0  // Initailzation when program is loaded

   ? footer_text + "Page " + Str (++page,3)
   Eject
Return Nil
```

รูปที่ 2 แสดงฟังก์ชัน page-footer กรณีที่เขียนในโปรแกรมคลิปเปอร์ 5.2

```
Function page_footer
   If Type ("page") = "U"
      Public page
      page = 0
   Else
      page = page + 1
   Endif
   ? footer_text + "Page " + Str (page,3)
   Eject
Return

Function page_set
   Parameter new_page

   page = new_page
Return Nil
•
```

รูปที่ 3

สรุป

ในเรื่องของตัวแปรในคลิปเปอร์ หากผู้เขียนโปรแกรมพยายามใช้ตัวแปรชนิดโลคัลและสแตติกแล้ว จะช่วยให้โปรแกรมที่เขียนมีการทำงานที่เร็วขึ้น นอกจากนี้ยังทำให้โปรแกรมมีความเป็นโมดูล่าร์ด้วย โดยทำให้โปรแกรมมีบัก (BUG) น้อยลง มีความน่าเชื่อขึ้น จัดการหรือทดสอบโปรแกรมก็ง่ายขึ้น อย่างไรก็ตาม ยังมีเทคนิคอื่น ๆ อีกที่จะช่วยให้โปรแกรมคลิปเปอร์ที่เขียนทำงานเร็วขึ้น โดยจะนำมาเสนอต่อไป

```
Static page := 0  // This External static variable
                  // Can use it through this file
                  // Initailize when program is loaded
                  // and when compile this file uses option /N
Function page_footer
   ? footer_text + "Page " + Str (++page,3)
   Eject
Return Nil

Function page_set ( new_page)
   page = new_page
Return Nil
•
```

รูปที่ 4

```
Function SetColor ( NewColor)
   Static ColorValue := "W/N"  // Initailize default color
                               // when program is loaded
   Local OldColor := ColorValue

   If NewColor != Nil
      ColorValue := NewColor
    Endif
    .
    .
    .

Return OldColor

•
```

รูปที่ 5